BIT Numerical Mathematics (2006) 46: 491–504
DOI: 10.1007/s10543-006-0076-x
Published online: 22 August 2006 – © Springer 2006

GENERATING NUMERICAL ALGORITHMS USING A COMPUTER ALGEBRA SYSTEM*

WALTER GANDER[1]

[1] *Institute of Computational Science, ETH Zurich, CH-8092 Zurich, Switzerland.*
email: gander@inf.ethz.ch

In memory of Germund Dahlquist (1925–2005).

Abstract.

An useful application of computer algebra systems is the generation of algorithms for numerical computations. We have shown in Gander and Gruntz (SIAM Rev., 1999) how computer algebra can be used in teaching to derive numerical methods. In this paper we extend this work, using essentially the capability of computer algebra system to construct and manipulate the interpolating polynomial and to compute a series expansion of a function. We will automatically generate formulas for integration and differentiation with error terms and also generate multistep methods for integrating differential equations.

AMS subject classification (2000): 65D25, 65D30, 65D32, 65L06.

Key words: computer algebra, numerical integration, numerical differentiation, multistep methods.

1 Newton–Cotes quadrature.

Newton–Cotes quadrature formulas are based on the following principle: to compute an approximation of the definite integral

$$\int_a^b f(x)\,dx$$

the function f is evaluated at $n+1$ points and interpolated by a polynomial of degree n. Then an approximation of the integral is obtained by integrating the polynomial instead of f.

If we take function values $f_i = f(x_i)$ at equidistant points, spaced $h = x_{i+1} - x_i$ apart, and consider the integration interval $(0, nh)$ then the approximation becomes

$$\int_0^{nh} f(x)\,dx \approx \int_0^{nh} P_n(x)\,dx = h\sum_{i=0}^{n} w_i f_i.$$

* Received February 3, 2005. Accepted September 15, 2005. Communicated by Per Lötstedt.

The weights w_i are usually computed by representing the polynomial in Lagrange form and integrating the *Lagrange polynomials*. However, here we do not need to specify the representation of the interpolating polynomial – we can concentrate just on the above mentioned principle.

In the literature *closed* and *open Newton–Cotes rules* are known. While closed rules use also the end-points of the integration interval, open rules use only the interior points.

1.1 Closed Newton-Cotes.

The following Maple function computes closed rules. We use $n+1$ equidistant points, including the end-points:

```
closedcotes := n -> factor(int(interp([seq(i*h, i=0..n)],
                [seq(f(i*h), i=0..n)], z),z=0..n*h)):
```

Simpson's rule is obtained by `closedcotes(2)`; and Milne's rule is generated with

```
milne := closedcotes(4);
```

$$milne := \tfrac{2}{45}\, h(7\, f(0) + 32\, f(h) + 12\, f(2\, h) + 32\, f(3\, h) + 7\, f(4\, h)).$$

The 9-points rule used for the function `quad8` in earlier versions of Matlab (R11) is obtained by

```
oldmatlabrule :=  closedcotes(8);
```

$$\begin{aligned}&\tfrac{4}{14175}\, h\,(+989\, f(0) + 5888\, f(h) - 928\, f(2\, h) + 10496\, f(3\, h)\\ &-4540\, f(4\, h) + 10496\, f(5\, h) - 928\, f(6\, h) + 5888\, f(7\, h) + 989\, f(8\, h)).\end{aligned}$$

1.2 Error of Newton–Cotes rules.

J. F. Steffensen proved in [4] that for sufficiently smooth functions f the discretization error of Newton–Cotes quadrature rules has the form

$$E = K f^{(m)}(\xi) h^{m+1}$$

with some ξ in the integration interval and constants K and m which depend only on the number of points and not on the integrand.

We can obtain the two constants K and m by simply expanding the error in a Taylor series:

$$E(h) = \int_0^{nh} P_n(x)\, dx - \int_0^{nh} f(x)\, dx.$$

As an example we compute the error of Milne's rule:

```
> milne := closedcotes(4);
> err := taylor(int(f(x), x= 0..4*h)-milne, h=0,8);
```

We obtain

$$\text{err} := -\tfrac{8}{945}\,(D^{(6)})(f)(0)h^7 + O(h^8), \quad \text{thus } m = 6 \text{ and } K = -\tfrac{8}{945}.$$

In Abramowitz-Stegun [1] Milne's rule is called Bode's rule (Formula 25.4.14) and the error is given by $E = -\frac{8}{945} f^{(6)}(\xi)h^7$ with $\xi \in (0, 4h)$. We can reproduce all closed Newton–Cotes formulas listed in [1] with the Maple function `closedcotes`.

We find a difference in the error constant of the 10-point formula (25.4.19). With our Maple function `closedcotes` we obtain

```
rule := closedcotes(9);
```

$$\begin{aligned}\text{rule} := \tfrac{9}{89600}\,h(&2857\,f(0) + 15741\,f(h) + 1080\,f(2\,h)\\ &+ 19344\,f(3\,h) + 5778\,f(4\,h) + 5778\,f(5\,h) + 19344\,f(6\,h)\\ &+ 1080\,f(7\,h) + 15741\,f(8\,h) + 2857\,f(9\,h))\end{aligned}$$

```
err := taylor(int(f(x),x=0..9*h)- rule, h=0,13);
```

$$\text{err} := -\tfrac{4671}{394240}\,(D^{(10)})(f)(0)h^{11} - \tfrac{42039}{788480}\,(D^{(11)})(f)(0)h^{12} + O(h^{13}).$$

Our error constant is $4671/394240 = 0.01184811282468$. However, in [1] the constant is $173/14620 = 0.01183310533516$. Since $4671 = 173 \times 27$ and $14620 \times 27 = 394740$ we conclude that in [1] there was a reading or transcribing error: the digit 2 in the denominator 394240 was interpreted as 7.

Also the constant in Formula 25.4.20 seems to be different. However, the constants are not really different – the fraction given in [1] is just unreduced.

1.3 Open Newton–Cotes rules.

We exclude the end-points in the Maple function `closedcotes` by replacing the range `i=0..n` with `i=1..n-1` and obtain

```
opencotes := n -> factor(int(interp([seq(i*h, i=1..n-1)],
                  [seq(f(i*h),i=1..n-1)], z), z=0..n*h)):
```

The error can again be computed with a Taylor series. For $n = 3, 4, 5, 6$ the statements

```
for i from 3 by 1 to 6 do
  rule := opencotes(i);
  err := taylor(rule - int(f(x), x= 0..i*h), h=0,i+4);
od;
```

produce the following rules:

rule	error
$\frac{3h}{2}\,(f_h + f_{2\,h})$	$\frac{3}{4}\,(D^{(2)})(f)(0)h^3 + \cdots$
$\frac{4h}{3}\,(2\,f_h - f_{2\,h} + 2\,f_{3\,h})$	$\frac{14}{45}\,(D^{(4)})(f)(0)h^5 + \cdots$
$\frac{5h}{24}\,(11\,f_h + f_{2\,h} + f_{3\,h} + 11\,f_{4\,h})$	$\frac{95}{144}\,(D^{(4)})(f)(0)h^5 + \cdots$
$\frac{3h}{10}\,(11\,f_h - 14\,f_{2\,h} + 26\,f_{3\,h} - 14\,f_{4\,h} + 11\,f_{5\,h})$	$\frac{41}{140}\,(D^{(6)})(f)(0)h^7 + \cdots$.

Comparing the results with those given in [1] we find a difference for the two point rule ($n = 3$). Our error constant is $3/4$ while in [1] the constant is given as $1/4$. It is not difficult to give a counter-example that shows that $1/4$ is wrong.

2 Generalizations of Newton–Cotes.

A second kind of open Newton–Cotes rules which are not so popular in the literature (see [6]) can also be derived easily. Open rules are handy for integrating functions which have a singularity at the end point like e.g. $\int_0^1 1/\sqrt{x}\,dx$.

Consider the same partition of the integration interval into n subintervals as for the closed rule: $x_i = ih$, $i = 0, \ldots, n$. Now we will use as nodes the *midpoints* of the intervals,

$$\frac{x_i + x_{i+1}}{2} = \left(i + \frac{1}{2}\right)h, \quad i = 0, \ldots, n-1.$$

Using the function

```
midopencotes := n -> factor(int(interp([seq((i+1/2)*h, i=0..n-1)],
                    [seq(f((i+1/2)*h), i=0..n-1)], z), z=0..n*h)):
```

We reproduce the two examples given in [6]:

```
rule :=  midopencotes(3);
```

$$\frac{3}{8}h\left(3\,f\left(\frac{1}{2}h\right) + 2\,f\left(\frac{3}{2}h\right) + 3\,f\left(\frac{5}{2}h\right)\right)$$

```
err := taylor(rule - int(f(x), x= 0..3*h),h,6);
```

$$\mathrm{err} := -\frac{21}{640}\,(D^{(4)})\,(f)\,(0)\,h^5 + O(h^6)$$

```
rule := midopencotes(5);
```

$$\frac{5}{1152}h\left(275\,f\left(\frac{1}{2}h\right) + 100\,f\left(\frac{3}{2}h\right) + 402\,f\left(\frac{5}{2}h\right) + 100\,f\left(\frac{7}{2}h\right) + 275\,f\left(\frac{9}{2}h\right)\right)$$

```
err := taylor(rule - int(f(x), x= 0..5*h),h,9);
```

$$\mathrm{err} := -\frac{5575}{193536}\,(D^{(6)})\,(f)\,(0)\,h^7 - \frac{27875}{387072}\,(D^{(7)})\,(f)\,(0)\,h^8 + O(h^9).$$

The formulas up to 6 points and the 8 points formula have all positive weights. The 7 points, 9 points and higher are less stable since they have positive and negative weights.

There is no need to use equidistant function values. Consider e.g. the Clenshaw–Curtis quadrature rules which are of Newton–Cotes type (with prescribed nodes) and which use Chebyshev nodes (zeros or extrema of the Chebyshev polynomials). Using the zeroes of the Chebyshev polynomial we obtain the function

```
clenshaw2 := n -> factor(int(interp( ([seq(cos(Pi/2/n +  k*Pi/n),
             k=0..n-1)]), [seq(f(cos(Pi/2/n + i*Pi/n)), i=0..n-1)],
             z), z=-1..1)):
```

The rule for $n = 4$ is

```
clenshaw2(4):
collect(%,f);
```

$$
\begin{aligned}
&+1/3\,\frac{(-3\,(\cos(1/8\,\pi))^2+1)f(-\cos(3/8\,\pi))}{(\cos(3/8\,\pi)+\cos(1/8\,\pi))(\cos(3/8\,\pi)-\cos(1/8\,\pi))}\\
&+1/3\,\frac{(3\,(\cos(3/8\,\pi))^2-1)f(-\cos(1/8\,\pi))}{(\cos(3/8\,\pi)+\cos(1/8\,\pi))(\cos(3/8\,\pi)-\cos(1/8\,\pi))}\\
&\quad 1/3\,\frac{(3\,(\cos(3/8\,\pi))^2-1)f(\cos(1/8\,\pi))}{(\cos(3/8\,\pi)+\cos(1/8\,\pi))(\cos(3/8\,\pi)-\cos(1/8\,\pi))}\\
&+1/3\,\frac{(-3\,(\cos(1/8\,\pi))^2+1)f(\cos(3/8\,\pi))}{(\cos(3/8\,\pi)+\cos(1/8\,\pi))(\cos(3/8\,\pi)-\cos(1/8\,\pi))}.
\end{aligned}
$$

The analytical expressions for the weights are complicated already for $n = 4$. Therefore we evaluate them numerically:

```
evalf(%);
```

$$
\begin{aligned}
&0.7357022609\,f\left(-\cos\left(3/8\,\pi\right)\right)+0.2642977395\,f\left(-\cos\left(1/8\,\pi\right)\right)\\
&\quad+0.2642977395\,f\left(\cos\left(1/8\,\pi\right)\right)+0.7357022609\,f\left(\cos\left(3/8\,\pi\right)\right).
\end{aligned}
$$

When computing directly with numerical values of the nodes we get the function

```
clenshaw := n -> factor(int(interp( evalf([seq(cos(Pi/2/n +
            k*Pi/n),k=0..n-1)]), [seq(f(cos(Pi/2/n + i*Pi/n)),
            i=0..n-1)], z),z=-1..1)):
```

Now we obtain for the 4 point rule:

```
clenshaw(4);
```

$$
\begin{aligned}
&0.7357022606\,f\left(-\cos\left(3/8\,\pi\right)\right)+0.2642977394\,f\left(-\cos\left(1/8\,\pi\right)\right)\\
&\quad+0.2642977410\,f\left(\cos\left(1/8\,\pi\right)\right)+0.7357022590\,f\left(\cos\left(3/8\,\pi\right)\right).
\end{aligned}
$$

Table 2.1: Clenshaw–Curtis quadrature rule.

+0.059555871355561788 9810|618352109	$f\left(-\cos\left(1/8\,\pi\right)\right)$
+0.059555871355561788 9810|194163814	$f\left(\cos\left(1/8\,\pi\right)\right)$
+0.145224707542664168068|395037185	$f\left(\cos\left(3/8\,\pi\right)\right)$
+0.145224707542664168068|446232731	$f\left(-\cos\left(3/8\,\pi\right)\right)$
+0.010758146483720551 9302|436197122	$f\left(\cos\left(1/40\,\pi\right)\right)$
+0.010758146483720551 9302|521694137	$f\left(-\cos\left(1/40\,\pi\right)\right)$
+0.037536816465188695 6526|442821896	$f\left(\cos\left(\frac{3}{40}\,\pi\right)\right)$
+0.037536816465188695 6526|671632191	$f\left(-\cos\left(\frac{3}{40}\,\pi\right)\right)$
+0.082463179919547056901|5809292332	$f\left(\cos\left(\frac{7}{40}\,\pi\right)\right)$
+0.082463179919547056901|4731335691	$f\left(-\cos\left(\frac{7}{40}\,\pi\right)\right)$
+0.101731206298401790 8900|93431821	$f\left(\cos\left(\frac{9}{40}\,\pi\right)\right)$
+0.101731206298401790 8899|66015632	$f\left(-\cos\left(\frac{9}{40}\,\pi\right)\right)$
+0.119653106977172112 6246|52776280	$f\left(\cos\left(\frac{11}{40}\,\pi\right)\right)$
+0.119653106977172112 6246|96669685	$f\left(-\cos\left(\frac{11}{40}\,\pi\right)\right)$
+0.133781854455624412 8800|70052834	$f\left(\cos\left(\frac{13}{40}\,\pi\right)\right)$
+0.133781854455624412 8800|15604605	$f\left(-\cos\left(\frac{13}{40}\,\pi\right)\right)$
+0.152680267016408376 40783|7523885	$f\left(\cos\left(\frac{17}{40}\,\pi\right)\right)$
+0.152680267016408376 40783|9590512	$f\left(-\cos\left(\frac{17}{40}\,\pi\right)\right)$
+0.156614843485711045 66359|3648382	$f\left(\cos\left(\frac{19}{40}\,\pi\right)\right)$
+0.156614843485711045 66360|8318416	$f\left(-\cos\left(\frac{19}{40}\,\pi\right)\right)$

Notice that the coefficients are not the same as before. Since the nodes are symmetric with respect to zero, the corresponding weights should be equal. However, the numerical computation is not stable, the results differ considerably for larger n. Fortunately, the symmetry helps us to identify the accuracy. The difference of corresponding values indicates which digits are affected by the finite arithmetic.

We have to use multiple precision to compute the weights correctly to some required tolerance:

```
Digits := 30;
clenshaw(20);
```

Comparing the corresponding values of the weights in Table 2.1 (the vertical bar separates the different digits) we see that we loose about 9–10 decimal digits in this example. But with `Digits:=30` the weights still have enough correct digits so that the quadrature rule can be used for computations in IEEE arithmetic.

3 Numerical differentiation.

Interpolation can also be used to derive formulas for differentiation: interpolate $n+1$ points $(x_i, f(x_i))$ with a polynomial $P_n(x)$. Differentiate k times and evaluate $P_n^{(k)}(x)$ for some $x = x_j$ to approximate $f^{(k)}(x_j)$.

We specialize to equidistant points, introduce the discretization step $h = x_{i+1} - x_i$ and evaluate $P_n^{(k)}(x)$ for $x = jh$ with $j \in [0, 1, \ldots, n]$. A change of variable $x' = x - jh$ is useful for computing the discretization error with the Taylor expansion for $x' = 0$. We obtain the following Maple procedure:

```
formula := proc(m,k,j)
# m+1 equidistant points -jh, -(j-1)h,..., 0, h, 2h,...,(m-j)h.
# k-th derivative,   evaluated at x = 0
  local i, p, dp;
  p := interp([seq(i*h,i=-j..(m-j))],[seq(f(i*h),i=-j..(m-j))],x);
  dp := diff(p,x$k);
  simplify(eval(dp,x=0));
end :
```

The well known symmetric difference formula for the second derivative is computed by

```
rule:= formula(2,2,1);
```

$$\text{rule} := \frac{f(-h) - 2\,f(0) + f(h)}{h^2}.$$

For the discretization error we we expand difference

```
err := taylor((D@@2)(f)(0)-rule,h=0,8);
```

and obtain

$$\text{err} := -1/12\,(D^{(4)})\,(f)\,(0)\,h^2 - \tfrac{1}{360}\,(D^{(6)})\,(f)\,(0)\,h^4 + O(h^6).$$

Thus we conclude

$$f''(0) = \frac{f(h) - 2\,f(0) + f(-h)}{h^2} - \frac{1}{12} f^{(4)}(\xi) h^2, \quad \xi \in [-h, h].$$

It is now easy to reproduce the whole page 914 (*Coefficients for Differentiation*) given in [1] by this Maple procedure. As an example we compute for $k = 2$ and $m = 4$ the coefficients and expand the error term

```
k:=2;
m:=4;
for j from 0 to m/2 do
   rule:= formula(m,k,j);
   err := taylor(rule-(D@@k)(f)(0),h=0,m+3);
end do;
```

We obtain

$$\text{rule} := -1/12\,\frac{-35\,f(0)+104\,f(h)-114\,f(2\,h)+56\,f(3\,h)-11\,f(4\,h)}{h^2}$$
$$\text{err} := \left(5/6\,(D^{(5)})(f)(0)h^3+\tfrac{119}{90}\,(D^{(6)})(f)(0)h^4+O(h^5)\right)$$
$$\text{rule} := 1/12\,\frac{11\,f(-h)-20\,f(0)+6\,f(h)+4\,f(2\,h)-f(3\,h)}{h^2}$$
$$\text{err} := -1/12\,(D^{(5)})(f)(0)h^3-\tfrac{19}{360}\,(D^{(6)})(f)(0)h^4+O(h^5)$$
$$\text{rule} := 1/12\,\frac{-f(-2\,h)+16\,f(-h)-30\,f(0)+16\,f(h)-f(2\,h)}{h^2}$$
$$\text{err} := -\tfrac{1}{90}\,(D^{(6)})(f)(0)h^4+O(h^5).$$

In [1], p. 883, we find also formulas for numerical differentiation where the derivative is not evaluated at an interpolation point but at $x = ph$ with variable p. A small change in the above procedure is sufficient to generate these formulas:

```
formulap := proc(m,k,j)
# m+1 equidistant points -jh, -(j-1)h,..., 0, h, 2h,...,(m-j)h.
# k-th derivative,   evaluated at x = p*h
  local i, pol, dpol;
  pol := interp([seq(i*h,i=-j..(m-j))],[seq(f(i*h),i=-j..(m-j))],x);
  dpol := diff(pol,x$k);
  simplify(eval(dpol,x=p*h));
  map(collect,%,f);
end :
```

We generate formula 25.3.4, 25.3.5 and 25.3.6 of [1] with the statements

```
formulap(2,1,1);
```

$$\frac{1}{2h}\left((2\,p-1)\,f\,(-h)-4\,pf\,(0)+(2\,p+1)\,f\,(h)\right)$$

```
formulap(3,1,1);
```

$$\frac{1}{6h}((-3\,p^2+6\,p-2)f(-h)+(9\,p^2-12\,p-3)f(0)$$
$$+(6\,p-9\,p^2+6)f(h)+(3\,p^2-1)f(2\,h))$$

```
formulap(4,1,2);
```

$$\frac{1}{12h}(+(2\,p^3-3\,p^2-p+1)f(-2\,h)+(16\,p-8\,p^3+6\,p^2-8)f(-h)$$
$$+(-30\,p+12\,p^3)f(0)+(8-8\,p^3+16\,p-6\,p^2)f(h)$$
$$+(-p+3\,p^2+2\,p^3-1)f(2\,h)).$$

A formula not displayed in [1] is e.g. the following symmetric expression for the second derivative

```
formulap(4,2,2);
```

$$\frac{1}{12h^2}[(-6\,p+6\,p^2-1)f(-2\,h)+(12\,p-24\,p^2+16)f(-h)$$
$$+(36\,p^2-30)f(0)+(16-24\,p^2-12\,p)f(h)$$
$$+(6\,p^2+6\,p-1)f(2\,h)].$$

4 Multistep methods for differential equations.

4.1 Adams-Bashforth.

Adams–Bashforth extrapolation methods are based on the following principle: to solve the differential equation $y' = f(x, y)$ approximately we integrate and approximate the integral by interpolation

$$y_1 - y_0 = \int_{x_0}^{x_1} y'(x)\,dx \approx \int_{x_0}^{x_1} P_{k-1}(x)\,dx \tag{4.1}$$

where P_{k-1} interpolates the derivative

x	$x_0,$	$x_{-1},$	$\cdots,$	$x_{-(k-1)}$
$y'(x)$	$y'_0,$	$y'_{-1},$	$\cdots,$	$y_{-(k-1)}$

.

In Equation (4.1) we assume that we already know the values of y_i and y'_i for $i = -(k-1), \ldots, -1, 0$ and thus we obtain an expression for the new value y_1. If again we use equidistant points $h = x_i - x_{i-1}$, make a change of variable $x' = x - x_0$ and use k points for the interpolation we obtain the following Maple function:

```
adamsbash := k -> y(h) = y(0)+
             factor(int(interp([seq(-i*h, i=0..k-1)],
             [seq(D(y)(-i*h), i=0..k-1)], z),z=0..h)):
```

Euler's method ($k = 1$) and the discretization error are computed by

```
adamsbash(1);
taylor(rhs(%)-y(h), h=0,3);
```

We get

$$y(h) = y(0) + \mathrm{D}(y)(0)h$$

and

$$-1/2\,(D^{(2)})(y)(0)h^2 + O(h^3).$$

The well known 4th order method is obtained by

```
adamsbash(4);
```

$$y(h) = y(0) + 1/24\, h(55\, \mathrm{D}(y)(0) - 59\, \mathrm{D}(y)(-h) + 37\, \mathrm{D}(y)(-2\, h) - 9\, \mathrm{D}(y)(-3\, h)).$$

The discretization error is computed by the expansion

```
taylor(rhs(%)-y(h), h=0);
```

$$-\tfrac{251}{720}\,(D^{(5)})(y)(0)h^5 + O(h^6).$$

Notice how easily we can generate the methods with the help of a computer algebra system. Compare this with classical approaches like e.g. [3] who were concerned with minimizing the work for hand computations. For that purpose Newton's interpolation formula for equidistant nodes is used

$$P(x) = \sum_{m=0}^{q} (-1)^m \binom{-s}{m} \nabla^m f_p, \quad s = \frac{x - x_p}{h}, \quad h = x_{i+1} - x_i$$

and

$$y_{p+1} - y_p = \int_{x_p}^{x_{p+1}} P(x)\, dx = h \sum_{m=0}^{q} \gamma_m \nabla^m f_p$$

with the coefficients

$$\gamma_m = (-1)^m \int_0^1 \binom{-s}{m}\, ds.$$

To compute the coefficients in an elegant way, Henrici [3] uses a *generating function*

$$G(t) = \sum_{m=0}^{\infty} \gamma_m t^m = \int_0^1 \sum_{m=0}^{\infty} (-t)^m \binom{-s}{m}\, ds = -\frac{t}{(1-t)\log(1-t)}.$$

The γ_m are obtained by expanding and comparing coefficients in

$$-\frac{\log(1-t)}{t} G(t) = \frac{1}{1-t}.$$

Finally we would like to have the coefficients $\beta_{q\rho}$ from

$$y_{p+1} - y_p = h \sum_{m=0}^{q} \gamma_m \nabla^m f_p = h \sum_{\rho=0}^{q} \beta_{q\rho} f_{p-\rho}$$

which requires the computation

$$\beta_{q\rho} = (-1)^\rho \left\{ \binom{\rho}{\rho} \gamma_\rho + \binom{\rho+1}{\rho} \gamma_{\rho+1} + \cdots + \binom{q}{\rho} \gamma_q \right\}.$$

4.2 Adams-Moulton.

Adams–Moulton multistep integration methods are implicit. The new point $y'(x_1) = f(x_1, y_1)$ is also used for constructing the interpolating polynomial $P_k(x)$. Therefore the unknown y_1 appears on both sides of Equation (4.1). The following Maple function arises from `adamsbash` by changing the range from `0..k-1` to `i=-1..k-2`:

```
adamsmoulton := k ->y(h) =  y(0)
                    + factor(int(interp([seq(-i*h, i=-1..k-2)],
                      [seq(D(y)(-i*h), i=-1..k-2)], z),z=0..h)):
```

The three point formula is generated by

```
adamsmoulton(3);
```

$$y(h) = y(0) + 1/12\, h\, (5\, \mathrm{D}(y)(h) + 8\, \mathrm{D}(y)(0) - \mathrm{D}(y)(-h)).$$

In usual mathematical notation this is:

$$y_1 = y_0 + \tfrac{1}{12} h \left(5 f(x_1, y_1) + 8 y_0' - y_{-1}'\right).$$

The discretization error is again obtained by a Taylor expansion

```
taylor(rhs(%)-y(h),h=0);
```

$$1/24\, (D^{(4)})(y)(0) h^4 + O(h^5).$$

4.3 General multistep methods.

Stoer and Bulirsch [5] consider multistep methods in the following general form:

$$y(x_{p+k}) - y(x_{p-j}) = \int_{x_{p-j}}^{x_{p+k}} f(t, y(t))\, dt.$$

Again we compute an approximation of the integral by interpolating the function by a polynomial $P_q(x_i) = f(x_i, y(x_i))$, $i = p, p-1, \ldots, p-q$. Without loss of

generality we could assume that $p = 0$, however, the index shift p is handy when comparing methods or generating predictor corrector methods.

For equidistant points $x_{i+1} - x_i$ we have four parameters to choose:

- the number of interpolation points defined by the degree q of the interpolating polynomial.
- j: the lower bound for the integral is $x_{p-j} = (p - j)h$.
- k defines the upper bound for the integral $x_{p+k} = (p + k)h$.
- the numbering of the points defined by the index shift p.

We generate the multistep methods including the discretization error with the following Maple function:

```
multistep  := proc(k,j,q,p)
 local i, IP, err, formel;
 IP := int(interp([seq((p-i)*h,i=0..q)],
       [seq(D(y)((p-i)*h),i=0..q)],x),x=(p-j)*h..(p+k)*h);
 formel := y((p+k)*h) -y((p-j)*h) = factor(IP);
 err:=taylor(lhs(%)-rhs(%),h=0,q+3);
 [formel,err]
end:
```

Using `multistep(1,0,q,0)` we obtain the explicit Adams–Bashforth methods. With `multistep(0,1,q,1)` implicit Adam–Moulton methods are produced. As an example we compute both fourth order methods (Formula 25.5.4 and 25.5.5 in [1]):

```
adams4:= multistep(1,0,3,0);
```

$$\begin{aligned}\text{adams4} := \Big[& y(h) - y(0) = -1/24\, h(-55\, \mathrm{D}(y)(0) + 59\, \mathrm{D}(y)(-h) \\ & - 37\, \mathrm{D}(y)(-2\, h) + 9\, \mathrm{D}(y)(-3\, h)),\ \left(\tfrac{251}{720}\, (D^{(5)})(y)(0) h^5 + O(h^6)\right)\Big]\end{aligned}$$

```
moulton4:= multistep(0,1,3,1);
```

$$\begin{aligned}\text{moulton4} := \Big[& y\,(h) - y\,(0) = 1/24\, h\,(+9\, \mathrm{D}\,(y)\,(h) + 19\, \mathrm{D}\,(y)\,(0) \\ & -5\, \mathrm{D}\,(y)\,(-h) + \mathrm{D}\,(y)\,(-2\, h))\,,\ \left(-\tfrac{19}{720}\, (D^{(5)})\,(y)\,(0)\, h^5 + O(h^6)\right)\Big].\end{aligned}$$

The predictor/corrector pair (Formula 25.5.14 in [1]) is generated by:

```
pred := multistep(1,5,4,0);
corr := multistep(0,4,4,1);
```

$$\begin{aligned}\text{pred} := \Big[& y(h) - y(-5\, h) = 3/10\, h(11\, \mathrm{D}(y)(0) - 14\, \mathrm{D}(y)(-h) \\ & + 26\, \mathrm{D}(y)(-2\, h) - 14\, \mathrm{D}(y)(-3\, h) + 11\, \mathrm{D}(y)(-4\, h)),\ (O(h^7))\Big] \\ \text{corr} := \Big[& y(h) - y(-3\, h) = \tfrac{2}{45}\, h(7\, \mathrm{D}(y)(h) + 32\, \mathrm{D}(y)(0) \\ & + 12\, \mathrm{D}(y)(-h) + 32\, \mathrm{D}(y)(-2\, h) + 7\, \mathrm{D}(y)(-3\, h)),\ (O(h^7))\Big].\end{aligned}$$

Nyström methods are generated by `multistep(1,1,q,0)`. The simplest one is the mid-point rule:

```
midpoint := multistep(1,1,1,0);
```

$$\text{midpoint} := \Big[y\,(h) - y\,(-h) = 2\,\mathrm{D}\,(y)\,(0)\,h,\ (1/3\,(D^{(3)})\,(y)\,(0)\,h^3 + O(h^4))\Big].$$

Finally for $q = 2, 3$ `multistep(0,2,q,0)` produces Milne–Simpson rules. E.g. for $q = 3$ we get

```
milnesimpson := multistep(0,2,3,1);
```

$$\begin{aligned}\text{milnesimpson} := \Big[&y(h) - y(-h) = 1/3\,h(\mathrm{D}(y)(h) + 4\,\mathrm{D}(y)(0)\\ &+ \mathrm{D}(y)(-h)),\ \big(-\tfrac{1}{90}\,(D^{(5)})(y)(0)h^5 + O(h^6)\big)\Big].\end{aligned}$$

4.4 Backward differentiation formula.

Backward differentiation formulas (BDF) are generated by the following principle: Interpolate $n+1$ function values y_i and approximate the derivative $y'(x_1)$ by $P_n'(x_1)$ thus replace the differential equation by

$$P_n'(x_1) = f(x_1, y_1).$$

The following Maple function is easily understood:

```
bdf := n -> simplify(h*eval(diff(interp([seq(i*h, i = -(n-1)..1)],
        [seq(y(i*h), i = -(n-1)..1)], x), x), x=h)=h*D(y)(h));
```

For $n = 4$ we obtain

```
bdf(4);
```

$$\tfrac{25}{12}\,y(h) - 4\,y(0) + 3\,y(-h) + 1/4\,y(-3\,h) - 4/3\,y(-2\,h) = h\mathrm{D}(y)(h).$$

The discretization error is computed by

```
Err:= taylor(lhs(%)-rhs(%),h=0);
```

$$-1/5\,(D^{(5)})(y)(0)h^5 + O\left(h^6\right).$$

It is well known that only the first six BDF-methods are stable. They can be generated by the following statements:

```
for i from 1 to 6 do
 bdf(i);
 taylor(lhs(%)-rhs(%),h=0,i+2);
od;
```

$$
\begin{aligned}
y_h - y_0 &= hf(h, y_h) \\
\tfrac{3}{2}y_h - 2y_0 + \tfrac{1}{2}y_{-h} &= hf(h, y_h) \\
\tfrac{11}{6}y_h - 3y_0 + \tfrac{3}{2}y_{-h} - \tfrac{1}{3}y_{-2h} &= hf(h, y_h) \\
\tfrac{25}{12}y_h - 4y_0 + 3y_{-h} - \tfrac{4}{3}y_{-2h} + \tfrac{1}{4}y_{-3h} &= hf(h, y_h) \\
\tfrac{137}{60}y_h - 5y_0 + 5y_{-h} - \tfrac{10}{3}y_{-2h} + \tfrac{5}{4}y_{-3h} - \tfrac{1}{5}y_{-4h} &= hf(h, y_h) \\
\tfrac{49}{20}y_h - 6y_0 + \tfrac{15}{2}y_{-h} - \tfrac{20}{3}y_{-2h} + \tfrac{15}{4}y_{-3h} - \tfrac{6}{5}y_{-4h} + \tfrac{1}{6}y_{-5h} &= hf(h, y_h).
\end{aligned}
$$

REFERENCES

1. M. Abramowitz and I. A. Stegun, ed., *Handbook of Mathematical Functions*, Dover Publications, New York, 1974.
2. W. Gander and D. Gruntz, *Derivation of numerical methods using computer algebra*, SIAM Rev., 41(3) (1999), pp. 577–593.
3. P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations*, Wiley, New York, 1962.
4. J. F. Steffensen, *On the remainder form of certain formulas of mechanical quadrature*, Skand. Aktuarietidskrift, 4 (1921), pp. 201–209.
5. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Texts in Applied Mathematics, vol. 12, Springer, Berlin, 2002.
6. A. Walther, *Zur numerischen Integration*, Skand. Aktuarietidskrift, 8 (1925), pp. 148–162.